

GECCOS – the new Monitoring and Control System at DLR-GSOC for Space Operations, based on SCOS-2000

C. Stangl¹, B. Lotko², M.P. Geyer³, M. Oswald⁴ and A. Braun⁵

German Space Operations Center (GSOC)

German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling / Munich, Germany

At DLR-GSOC, the German Space Operations Center, the Satellite Monitoring and Control System (MCS) originating from ESA-SCOS-2000 was adapted for the first time for the mission CHAMP, beginning from the year 2000. Since then a custom GSOC branch of this MCS is in active development, both with respect to mission-specific adaptations as well as multi-mission related, ultimately leading to GSOC's own MCS called "GECCOS" – the GSOC Enhanced Command- and Control System for Operating Spacecrafts. GECCOS, based on SCOS-2000 Release 3.1, represents a generic MCS and supports a broad set of scientific and commercial satellite platforms: CHAMP, GRACE, TSX (TerraSAR-X, TanDEM-X, PAZ), EnMAP, TET, SmallGEO (HAG-1, EDRS-C, H2Sat), Spacebus 3000 (COMSATBw 1&2), Eurostar 3000 (EDRS-A) and in future SWARM Bus (GRACE-FO). Additionally, GECCOS has the capability to act as MCS as well as Central Check-out System (CCS) so it is capable of supporting S/C projects from AIT phase until mission operations phases. This has been demonstrated in the context of the missions TerraSAR-X, TanDEM-X, PAZ, TET and BIROS. That approach offers significant advantages regarding inherent validation of the future S/C operational MCS, being compatible with the S/C database (in SCOS-2000 terms Mission Information Base, MIB) as well as with flight control procedures (FCP), already within early AIT and S/C checkout phases. This is a key driver for the use of GECCOS within SmallGEO platform based S/C operations as their CCS is also based on SCOS-2000 Release 3.1. kernel. The combination of CCS and MCS data handling kernels is an important paradigm which is also one of the key drivers for future MCS/CCS projects like the European Ground Systems Common-Core (EGS-CC), a project led by ESA. In this contribution we present the main adaptations and advantages GECCOS offers when compared to classical MCS like ESA SCOS-2000 and point out how it can fulfill the MCS requirements for upcoming Missions operated at modern control centers.

I. Introduction: GECCOS as the DLR-GSOC generic Multi-Mission MCS with CCS capabilities

GECCOS is based on ESA's MCS development SCOS-2000 Release 3.1 and its story began 1999/2000 as the DLR-GSOC SMCS (Satellite Monitoring and Control System). The goal was to replace various legacy systems with one unique system suitable for current and future missions. Following this approach, several adaptations for upcoming missions have been incorporated, driven by GSOC in-house development and maintenance. To give an overview of its evolution Figure 1 shows the major steps of this development.

¹ Mission Control and Data handling System Team Member, German Space Operations Center (GSOC), German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V.), / DLR, Christian.Stangl@DLR.de

² Software Consultant, Freelancer, Austria

³ Mission Control and Data handling System Team Member, GSOC / DLR, Michael.Geyer@DLR.de

⁴ Software Consultant, Siemens Convergence Creators, Austria

⁵ Mission Control and Data handling System Team Lead, GSOC / DLR, Armin.Braun@DLR.de

A. Historical evolution

Starting point was the MCS SCOS-2000, Release 2.3, developed by ESA. Its first mission-specific adaptation was CHAMP, which was integrated end of 2001 and led to a hybrid Linux/SOLARIS system. After successful verification and validation of this system in flight it was further developed for the GRACE mission by GSOC. Although for GRACE, GSOC's legacy MCS was used in flight operations, these changes were valuable contributions to the evolution of the SMCS based on SCOS-2000. Some of these changes have been re-reported to the last major release of SCOS-2000, in particular SCOS-2000, Release 3.1, W03, which was driven by ESA-ESOC with contribution of DLR/GSOC. Since that time the evolution of GSOC's MCS, resulting in GECCOS, was done within a purely DLR driven branch.

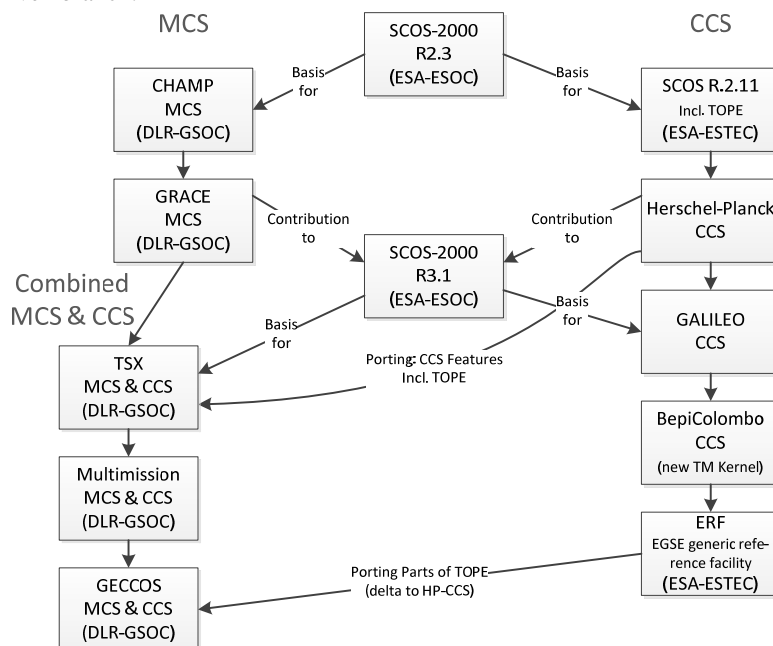


Figure 1 – Historical Evolution of GECCOS, based on ESA SCOS-2000

The adaptation for TerraSAR-X represented a major milestone due to two aspects: (i) in close cooperation with the spacecraft manufacturer, former Airbus company, Astrium Satellites Friedrichshafen, it was possible to consider the on-board data-handling and data-processing specifics within the MCS for later spacecraft operations. This goes hand in hand with (ii) the second aspect, that the same DLR SMCS was developed to serve as MCS as well as CCS for AIT purposes. The advantages are not only that AIT and operations use the same database, but also that the future MCS for S/C operations was validated during checkout phase. Since this stage of development a scripting and automation environment, based on TOPE, is integrated within the SMCS as well as many external interfaces (EXIF). This junction to MCS kernel-sided processing tasks was and is used during AIT of TanDEM-X, TET, PAZ and BIROS. Additionally, these add-ons were valuable contribution for interfacing outsided components as well as to automated tasks during S/C operations.

B. Generic Multi-Mission MCS with CCS capabilities

So far, DLR-SMCS evolved into a multi-mission MCS mapping the on-board data handling on-ground for a broad set of satellite platforms, as CHAMP, GRACE, TSX (DLR missions TerraSAR-X, TanDEM-X, PAZ (LEOP only)), TET (TET, BIROS) EnMAP (based on OHB's SAR-LUPE) and Spacebus 3000 (COMSATBw-1, COMSATBw-2); in recent times support for the new SmallGEO platform by OHB (DLR missions H2Sat, EDRS-C, HAG-1 (LEOP only)) and for AIRBUS D&S E3000 S/C BUS (EDRS-A) has been added. Future use is also planned for DLR missions EuCROPIS and GRACE-FO. The S/C are all CCSDS compatible, partially ESA-PUS compatible, partially with frame-based telemetry. Additionally, it was used successfully within studies like AITS (Advanced Integration and Test Services, Project with Airbus / former Astrium ST) and MICCRO (study regarding robotic operations in space) and was evaluated as possible MCS for Columbus, the European contribution at ISS (note: all mentioned missions are/have been operated by DLR-GSOC).

The system also profited from the CCS add-ons, as the CCS External Interfaces (EXIF) offer connections to MCS-external processes like GSOC-own display system SATMON⁷ (in use for all missions) including its capability of long-term mission archive, or connections to systems for automated commanding (used for TerraSAR-X and TanDEM-X¹, planned for EDRS-A, EDRS-C). Additionally, GECCOS is object of a study for acting as Offline Processing System for H/K Dump Data TM.

It has to be pointed out that mission- and BUS-specific features have been implemented based on one main branch (DLR/GSOC Multimission Satellite-MCS) such that they can be configured for supporting different S/C platforms. This is one of the most important requirements for GSOC, to avoid diverging code branches, i.e., bugfixes found within the Mission Operations Segment (MOS) of one mission are also applied to other missions, if they do not originate within deeply mission specific adaptations. Hence, the current version of GECCOS forms always the MCS basis of all current missions at DLR: missions in their operational phase just as well as missions prepared for launch, and likewise upcoming missions. The difficulties within this approach is discussed within a contribution of GSOC's software partner SIEMENS².

C. Modernization: Evolution to GECCOS

The last big milestone within the development of a GSOC-specific MCS is comprised of several stages of modernization (i) to get rid of legacy components and (ii) to be prepared for future operations and future missions within the next upcoming years. In chapters II(ff) we want to emphasize its advantages in comparison to previous systems.

D. Integration of GECCOS within DLR GSOC

In the following, it shall be illustrated how GECCOS, although subject of continuous maintenance and enhancement, is successfully integrated within many DLR GSOC's Mission Operation Segments (MOS) for missions mentioned above. Therefore a typical MOS setup is depicted in Figure 2.

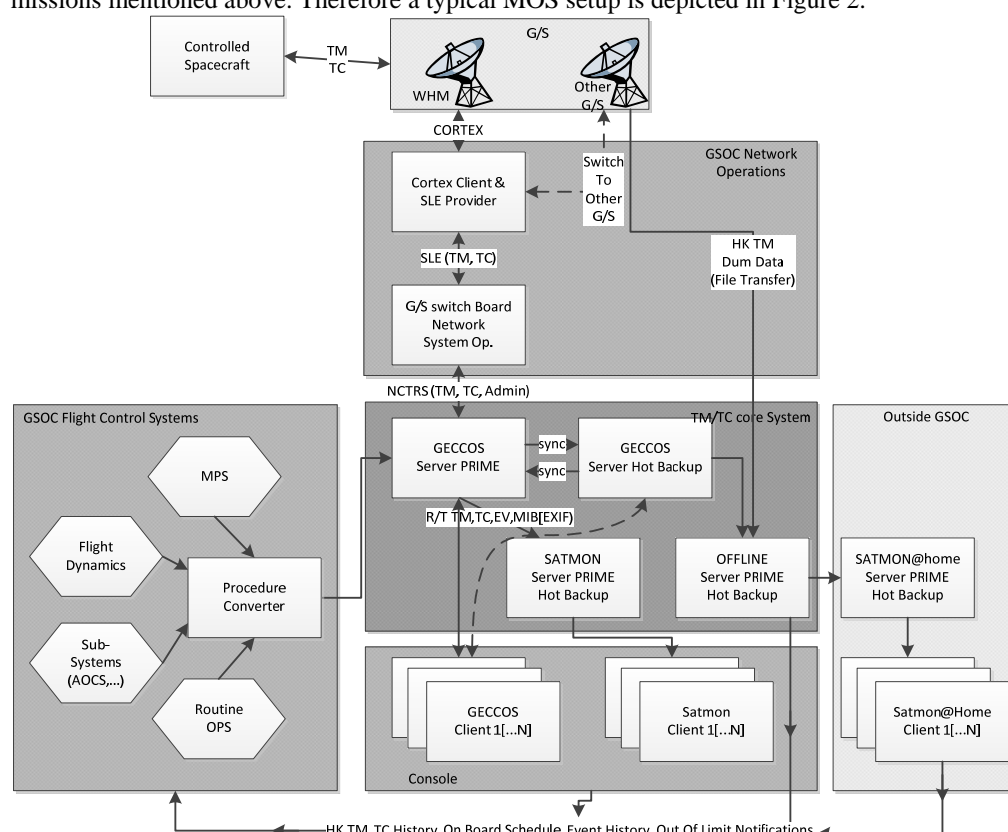


Figure 2 – Typical Setup of GSOC Mission Operations Segment with deeply integrated GECCOS Clients & Server

GECCOS, following a server / client setup, is forming the heart of all R/T data handling operations. It gets R/T data using the NCTRS I/F to GSOC own network operations which can switch via the SSB (SLE Switch Board)

between different certified G/S, connected via SLE (Space Link Extension). In the next step R/T data are received at the concerning G/S and in following parsed by the GECCOS TM chain. Additionally they are archived following online complete principle, and distributed following the online timely principle to all connected TM applications running on activated GECCOS clients. These can also retrieve data from the archive. TC data in turn are instantiated or loaded at clients' manual stacks ([I]MSTK, see also chapter I) and dispatched to the server sided TC chain; there they are processed and encrypted, where applicable, and finally routed to Network Operations/SSB, and their connected G/S for radiation, respectively. The MCS, GECCOS, is - also as many other systems - built up in a redundant manner (Prime / Hot Backup) whereat both servers synchronize each other to be ready for taking over operations. Additionally, within the TanDEM mission GECCOS is integrated such that two satellites can be operated simultaneously (TSX, TDX).

Many other MOS Subsystems like Mission Planning System (MPS, mission timeline generation), Flight Dynamics (maneuvers), Scheduling, S/C flight operations (Flight Control procedures for AOCS, PTS, ...), etc., can send their input to GECCOS using a dedicated XML schema mapping certain Flight Control Procedures (FCP), which are controlled, developed and versioned within the tool MOIS (done off-line). Their input is translated to the GECCOS Command Stack input format and can be dispatched automatically¹ or manually by command operators. The automated dispatching uses the scripting language TOPE, originating from the use of GECCOS as CCS. On the other side, some missions use Flight Control Procedures integrated directly within the S/C database. This possibility is provided by the Mission Information Base (MIB) format, the ESA SCOS-2000 database standard for S/C, operated by SCOS-2000 (based) systems.

Lots of other tools and Systems present within the MOS are connected to GECCOS via dedicated I/F, partially using EXIF's (external I/F's for TC, TM, event messages, S/C database/MIB entities) which have originally been developed for AIT activities. Hence, the R/T TM is routed to GSOC's own display system SATMON on parameter basis, which in turn also does archiving on parameter level. It distributes the processed data to clients or tools installed outside the MOS LAN (e.g. at home for on-call operations). After passages, processed data are transferred to the Offline Processing System which also does process HK dump data, recorded at G/S. This combined processed data in turn is distributed to various tools and subsystems for further analysis and/or processing and closes the loop e.g. for MPS, Flight dynamics and other subsystems like AOCS, etc.

II. Enhancements, new features and modernization approach

A key driver for the steady development of DLR's MCS leading to GECCOS are lots of enhancements and bug fixes triggered by engineering change requests (ECR) and non-conformance reports (NCR), a process followed since the early SCOS Release 2.3 days. Following table shall give an overview of changes, done since that time. Statistics have been added to reflect the complexity of the DLR implementations. To illustrate which effort it would take to build a MCS for Satellite missions operations from scratch, also cost estimations are shown. Latter results were derived using the tool SLOCCount by David A. Wheeler⁴. Due to these enhancements the MCS evolved to a well flight-proven software, stable and reliable for critical operations. This big advantage, the experience and comparison with high costs of a new system on the other hand are the reason why GSOC decided to keep at this system for the next 5-10 years.

Changes	Amount
Major enhancements	210
Major fixes	289
Statistics, DLR Multimission GECCOS without new GUI, with legacy ILOG Views	
Total Physical Source Lines of Code (SLOC) (Basis: Multimission DLR GECCOS Kernel on basis of SCOS-2000, i.e. without new GUI parts, for better comparison with ESA SCOS; (ESA SCOS: 507398)	661726
Development Effort Estimate, Person-Years (Person-Months) (Basic COCOMO model, Person-Months = $2.4 * (KSLOC^{**}1.05)$)	183.12 (2,197.47)
Schedule Estimate, Years (Months) (Basic COCOMO model, Months = $2.5 * (person-months^{**}0.38)$)	3.88 (46.54)
Estimated Average Number of Developers (Effort/Schedule)	47.21
Total Estimated Cost to Develop (average salary = \$56,286/year, overhead = 2.40).	\$ 24,737,382

Nr. of changed Lines	346991
Changes	52.4%
New Lines	24,3%
Statistics, New Gui Parts (Qt project)	
Total Physical Source Lines of Code (SLOC)	111654
Statistics, New Gui / Kernel sided parts ("BODY", see chapter I,ff)	
Total Physical Source Lines of Code (SLOC), ILOG free	635104

Table 1 – Changes by DLR within ESA SCOS-2000 since Release 3.1/W03 (last release done with DLR contribution). For better comparison (i) for version with standard ILOG Views based GUI and (ii) with DLR-driven GUI development, based on Qt (see chapter I,ff).

E. Additional features

In parallel to above mentioned adaptations for supporting different S/C platforms new features have been developed and integrated to come along with new operational requirements and experiences as well as needs on the developer and maintainer side. Not only space operations are done at GSOC, but also the MCS development is done in-house together with S/W contractors. This existing shortcut between S/C operations and software development results in short development cycles, lower costs and, particularly, a stringent validation approach for flight operation usage (see [2]). Following list gives an overview of implemented key features with respect to the original version of ESA-SCOS.

Topic	Corresponding changes
<i>Security features</i>	Password security / encryption
	TOPE Server whitelisting for avoiding TOPE access of unauthorized workstations
	Optional use of external TC encryption modules
	Study for future mission: secure data transmission between hosts (clients & redundant servers), based on ssh, to separate platform operations and hosted payload operations
<i>Interfaces</i>	Multiple connections to TM & TC gateways, following NCTRS protocol (TM & TC connection to the Ground-Station core network, connections to simulators, including redundant hosts)
	External Interfaces (EXIF) for AIT and operation purposes, e.g. connection to automated systems for autonomous S/C operations
	EXIF: close connection to MCS external system, e.g. the DLR display System SATMON; this includes access to TC, EV and TM History as well as Command Stacks which have to be sent, even via remote data transfer e.g. during on-call shift for monitoring at home
	Deployment of TOPE enhancements of CCS-located SCOS branches up to most evolved CCS by ESA/ESTEC, the ERF, incl. adaptations on the EXIF layer and necessary kernel changes; these include:
	Wrappers of EXIF layer for use within scripting engine (TOPE, based on tcl/tk), e.g. for autonomous S/C checkout procedures
	Direct link between flight operation procedure supervising tool (MOIS) and automated procedure execution via export of scripted procedure in TOPE scripting language
	Generic SCOE framework, reusable e.g. for POWER SCOE, KMF (key management facility, connected as SCOE) or as standalone TM generating tool for e.g. system monitoring
<i>Operational features</i>	Rework of SCOS On-board queue model (OBQM) including status & ACK updates
	MSTK improvements (IMSTK, colored stack midgets, TC History integration, procedure visualization)
	Improved Limit checking including soft and hard limits on textual calibrations
	Possibility of comprehensive patching of the S/C database (MIB) during run-time
	TC Acknowledge History (TCAH) and On-board Event History (OBEH, for LEO Satellites)
	Post-Pass Product function for fast TM product deployment after a passage of LEO Satellites
	Command Stack Supervisor function: Manual Stack visible within external tools like TM display System SATMON (note: also remote access, e.g. during on-call operations)
	Manual Stack offering possibility of comments and breakpoints, persistent within TC archive, for enhanced grouping and interleaving of TCs originating from different flight operation procedures

	Telemetry Checks and Telemetry configuration checks
<i>S/C Data-handling related</i>	Support of higher MIB ICD 6.9, like used within recent releases of ESOC SCOS-2000, Release 5
	Support of NEO MIB ICD 12.X, like used within EUTELSAT NEO
	TC randomization
	TC authorization
	Improved Variable Packet Processing
	Multiple time stamping
	Time correlation function
<i>Maintenance, S/W development related</i>	Use of various functions/tools from command line
	Improved configuration approach
	Use of git for revision control
	Re-build of MCS-internal sys-logging
	Performance features (Combined Command Handler)
<i>Operating System</i>	Novell SLES 11 / SP 3 (Linux Operating System)
<i>Toolchain</i>	C++ (programming language)
	Python, Tcl/Tk (scripting language)
	git (revision control and source code management (SCM) system)
	GCC 4.3 (compiler)
	waf (Python based tool for build automation, compatible with Continuous Integration tool JENKINS)
	Jenkins (Continuous Integration)
	Eclipse (IDE GECCOS Kernel sided)
	Qt Creator (IDE GUI sided)
	Qt commercial (application framework)
<i>Libraries, COTS</i>	Boost, Poco, C-Tree
	D-Bus, D-Bus GLib (Middleware for Kernel-GUI communication)
	Omniorb (CORBA Middleware)
	Squish for qt (automated testing tool, instrumenting Python based testing scripts)
	Squish coco (Code coverage tool GUI sided), lcov (Code coverage tool kernel sided)

Table 2 – Selection of DLR-GSOC driven enhancements on basis of SCOS-2000 R.3.1/W03 for GECCOS.

F. Performance of the GECCOS TM- and TC Chain

The Monitoring and Control System (MCS) of spacecraft operation centers like GSOC has to fulfill suitable performance requirements for uplinking TCs and downlinking housekeeping and payload data. Future missions are often a key-driver of increased performance requirements on the MCS as for example the contact frequency can be reduced with higher up- and downlink performance of the data-handling systems on-ground, which leads to less operations and not least reduced costs. For this reason a study was conducted to obtain a performance figure for TM- and TC capability of the GECCOS system. Following special preconditions on TC side - i.e. combining a block of TCs for "single-threaded" Manual Stack (MSTK) operations while ignoring all other possible MSTK inputs - a data rate of about 350 TCs/sec could be achieved. Considering TCs of mixed lengths this results in 75 kBytes/sec (standalone), run on a quite matured hardware. Together with the GSOC owned ground station WHM (Weilheim) and with enabled pre-transmission verification option (operational BD [bypassed] mode⁵) the data rate reaches saturation at about 20 kBytes/sec (~100 mixed-length TCs/sec), whereas the reduction of datarate is caused by handshakes between GECCOS and G/S software (sliding window of 10 TCs for uplink verification), necessary for a reliable uplink verification. This is a speed-up of factor 5k relative to a TC intensive actual mission like TerraSAR-X/TanDEM-X. On TM side the telemetry processor (SCOS-2000 descendent but tuned "Packetizer, TPKT") was

measured processing at an update rate of ~4500 TM packets per second or about 2.5MB/sec, and 4500 updates/sec, respectively, for a TM parameter incl. all processing overhead as e.g. out-of-limit processing, etc.

G. Modernization approach

To come along with future requirements and future maintenance aspects, DLR decided to modernize its own S-MCS, based on SCOS, which finally led to the actual GECCOS. As also smoothly maintenance for long-lasting missions should be guaranteed, lots of legacy "*Commercials Of The Shelf*" (COTS) had to be sorted out.

First of all the originally used visualization library, *ILOG VIEWS*, has to be mentioned, used at SCOS-2000 until ESA Release 5. This library is also used for threading techniques, what makes an exchange very hard (see chapter III). *ILOG VIEWS* was originally developed by ILOG, a company purchased and incorporated into IBM. However, the library used for SCOS UI was not further developed which led to problems in using modern S/W development techniques (modern compiler & debugger, automated testing). It prohibits the use of more modern compilers as it is only available for GCC 3.3.X. It can be linked with the former used GCC 3.3.6 but not with versions 3.4 upwards. Current versions of Linux distributions (including Novell SLES11) are GCC 4.2 and upwards. Additionally, single workstation licenses are still quite expensive.

This fact makes it even harder to deliver and maintain a product capable for long-lasting mission operation phases. Together with the decision to enforce automated S/W testing and validation, DLR decided to replace the Graphical User Interface (GUI) with Qt, which is heavily used within LINUX desktop KDE, and which has a strong developer community, especially due to its open-source branch. Additionally, Qt implies a commercial line which offers professional developer support. Another advantage of Qt is the possibility of automated testing using froglogic's System *SQUISH* for Qt (see chapter K).

DLR already conducted studies for the replacement of other COTS, which is or will be in work soon. On the one side *CTREE* has to be mentioned which is in use in ESA SCOS-2000 R.3.1 for the archiving system. Since this system has several technical limitations and as it has a commercial/ESA provided license, *CTREE* will be removed with a suitable archive technology (e.g. MySQL like for SCOS-2000 R.5 or SQLite as done at ESA/ESTEC "EGSE reference facility" (ERF)). Another legacy COTS, still in use within most SCOS-based MCS like NEO/hifly or CCS like the ERF, is *POST++* which handles the MIB storage. DLR tends to replacement since it has a number of drawbacks: (i) It needs a fixed address space. Newer kernels perform a randomized mapping of process address spaces to make it harder for viruses to attack which has to be switched off when using *POST++*. (ii) Another drawback is that it rather hampers debugging, since convenient tools and memory debuggers don't work when entering code where *POST++* is used. (iii) Every time the MIB is changed SCOS/*POST* based systems have to be restarted; (iv) all applications have to be linked with the libraries in the same order, otherwise strange crashes can appear; (v) after recompilation of a subsystem which provides *POST* persistent classes, SCOS has to be shutdown, and a new import has to be done followed by a startup. Due to those drawbacks DLR/GSOC will replace the COTS *POST++*, still present within GECCOS, with *boost*. As *boost* version 1.45 (current: 1.55) is the latest one supported by GCC 3.3.6, and which in turn had to be used in combination with ILOG, this work can be started only now, where first ILOG-free versions of GECCOS have been finished. Last but not least, GSOC conducted a study to detect COTS products with too restrictive open source licenses like *GPL*, which are to be replaced, too.

III. Architecture of GECCOS with respect to new GUI components

H. GECCOS Architecture for Kernel-GUI separation

The replacement of the obstructive library ILOG was the most significant milestone for GECCOS so far. The decision about the system architecture – always difficult and with extensive consequences – was a very complex issue concerning the hook-up of new GUI parts for the new GECCOS system replacing ILOG Views based code. The starting position, DLR driven branch on basis of SCOS-2000 R.3.1, was an outdated and rather difficult to port system which could only be built with an obsolete compiler (GCC 3.3.X); therefore updating the OS against new LINUX distributions is always a challenge for SCOS-2000 R.3.1 based systems, in general, that still come along with the originally ILOG VIEWS based UI and threading techniques. On the other side, when considering a complete new development of the MCS, the DLR/GSOC SCOS-2000 based S-MCS has one huge advantage: it works stable, carries 10 years of bug fixes, improvements and adaptations, is well validated in flight and coped by the operating staff.

Therefore, instead of starting from scratch and writing a whole new system, we decided to modernize the existing one without loss of its functionality and its look and feel of graphical user interfaces (GUIs). The obvious solution was to keep the old functional software modules, and to replace the obsolete part, in this case the GUI. In our case however, such a solution poses additional difficulty: the old system could be compiled only with GCC

3.3.X versions (most recent compiler gcc-3.3.6) whereas the modern GUI software, to be built with Qt, needs at least the GCC-4.x environment.

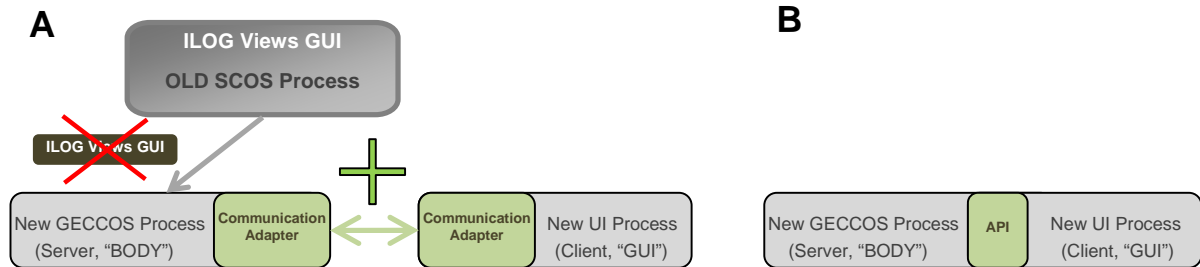


Figure 3 – A: Transformation of the old SCOS-2000 based system. Former SCOS-2000 based applications had deep integrated ILOG Views GUI and threading technique. On their basis, legacy code parts of ILOG Views were eliminated and processes transformed into a new server- & kernel-sided process called "BODY"; this is completed by new coded "GUI" process. Both are carrying a proper communication adapter; **B: Assembly of new GUI related client and server processes for GECCOS.**

An additional requirement was that new UI's – e.g. an overworked Manual Stack – shall outlast the SCOS-based kernel to be capable for later re-use during integration with a future MCS kernel; this is planned in the frame of future integration of the upcoming "European Ground Segment - Common Core" (EGS-CC), a project led by ESA³ with the goal to harmonize between industry and agencies, where DLR is one of its stakeholders.

These constraints enforced the decision to follow a UI – kernel separated approach for better future re-use the solution was to follow a client – server architecture as depicted within Figure 3 A. This allows (i) still to use the stable, well flight-proven parts of pre-GECCOS, which carry the functionalities – like a MCS kernel – in a server role; this server is (ii) complemented with the new written GUI as the controlling client. Each major process, i.e. the "server sided" part, is called BODY, the corresponding new UI part "GUI".

In a first project verification phase, the client (GUI) and server (BODY) had only to be compiled with different compilers (3.3.6 for "BODY", current LINUX installation 4.x compiler for "GUI") and ran in different environments (different LD_LIBRARY_PATH environment variable setting). After this verification phase, the "BODY" source code was ported to the 4.x GCC compiler, and the decision whether the client–server architecture shall be kept, or whether the "GUI" and "BODY" parts shall be connected via an API could be made. Following this clear definition of the interface between the "BODY" and "GUI" parts, which is necessary for the client-server approach, it is always possible to assemble client and server parts via a programming API, as shown in Figure 3 B.

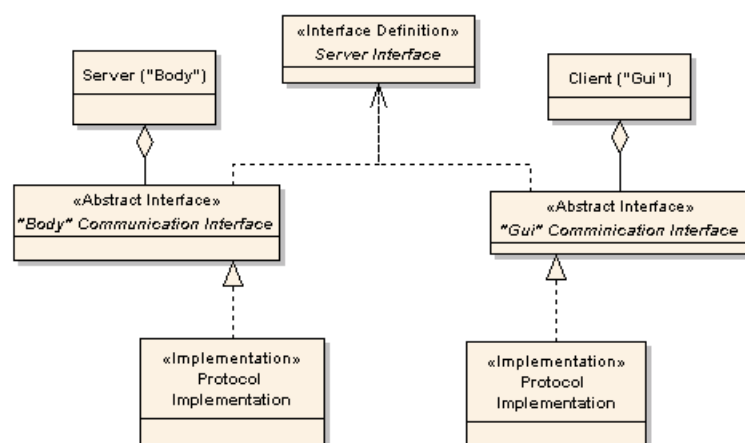


Figure 4 – Communication Architecture for GECCOS Client-Server approach

To summarize, this client- server solution for the hook-up of the new Qt based GUI parts of GECCOS has following advantages: (i) scalability, (ii) high portability, (iii) possibility of remote control (iv) clear interface

separation and (v) a better testability (see chapter K) and nevertheless (vi) the possibility of later re-use, e.g. in the frame of future integration of the upcoming EGS-CC. On the other side, of course, also the disadvantages have to be mentioned like (a) enhanced network traffic and (b) enhanced development and (c) maintenance effort. In the current phase (as described in sections below), the advantages of the client-server solution outweigh the disadvantages, which enforce the client-server architecture.

I. GECCOS GUI: Communication Protocol

As the envisaged architecture shall ensure greatest possible flexibility in application of communication protocol, the following solution was implemented as illustrated within Figure 4. The abstract interface defined for both, the server (BODY) and the client (GUI) allows implementation of any IPC protocol like D-Bus, CORBA, DCE etc. It also allows a future replacement of an already implemented protocol. The last decision to be made was related to the choice of the communication protocol. As SCOS-2000 R.3.1 based systems already use CORBA protocol, this should be the easiest and the most obvious solution. However, our final decision was using D-Bus for the following reasons.

i. D-Bus versus CORBA

D-Bus was originally developed for LINUX Inter-Process Communication (IPC) and Remote Procedure Calling (RPC) mechanism and has been designed to allow communication between system-level processes and user processes. The usual case is "many to many" or "one to many" communication through a central server application called the "bus"; but also peer to peer and socket based TCP host to host communication is possible. D-Bus daemon is a part of every LINUX installation, and has numerous language bindings C++, Java, Python, TCL. The most important C++ implementations are GDBus (D-Bus support in GLib) and QtDBus - the D-Bus implementation in Qt. Unlike CORBA, D-Bus does *not* specify the API for the language bindings. Instead, "native" bindings adapted specifically to the conventions of a framework such as QObject, GObject, C#, Java, Python, etc. are encouraged⁶. The crucial advantage of D-Bus against CORBA is the direct support for "signals" as used within the GLib/Qt framework. In contrary to UDP, D-Bus signals are reliable and within one connection and for the same object it is ensured that they are to be delivered in the same order in which they were sent.

CORBA is an industry standard strongly supported by the *Open Management Group* (OMG) designed to support object-oriented IPC between objects. It supports marshaling of parameters and specifies the API for language bindings. Both GNOME and KDE have used CORBA but now D-Bus with all its features seems to be much better tailored for the IPC between processes within a session or within one system. To complete this short comparison it shall be mentioned, that many CORBA implementations are faster than the libdbus reference implementation.

Altogether, the decision fell for using D-Bus, as the most important requirement for the new system architecture was the possibility of sending reliable signals between kernel-sided parts (BODY) and GUI.

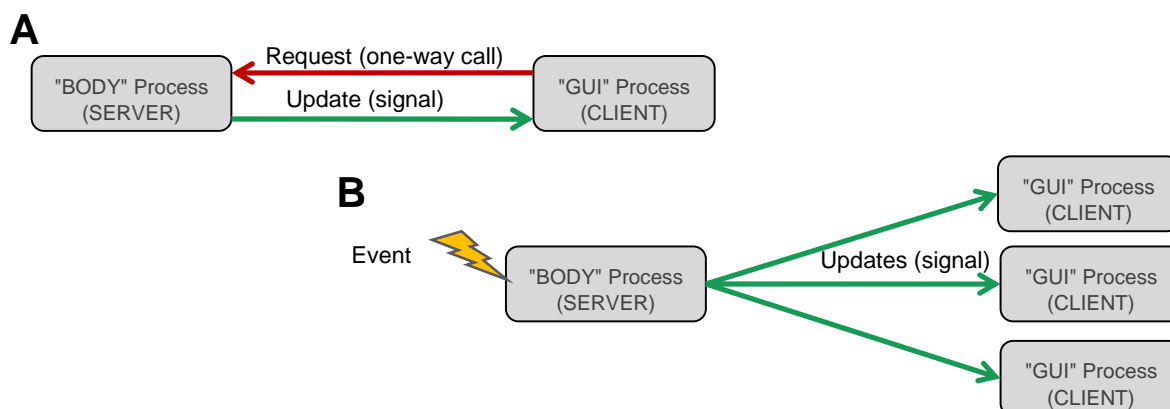


Figure 5 – A: GECCOS Communication using one-way calls Signals; B: GECCOS Communication, signal broadcast

ii. GECCOS architecture

Following this approach the new GECCOS architecture extensively uses the signaling capability of D-Bus. All events are propagated (broadcasted) as signals; most calls are implemented as one-way call. Whenever a "GUI" application is sending a request it expects a signal confirming the requested changes from "BODY". This approach ensures that the requested changes really have been performed by the server (BODY) side, and it allows a single

threaded implementation of client ("GUI") processes, without risk of hang-on by longer lasting operations (see Figure 5 A).

It has to be noticed that this technique also allows spontaneous updates when the server changes a status. As a result, also the simultaneous distribution/broadcast of a signal is possible, which in turn does not only update one client but multiple clients, as shown in Figure 5 B. This behaviour is scalable such, that signals sent from one D-Bus interface updates registered clients which only need to listen out, i.e. to subscribe on D-Bus to get an update. This feature can be utilized e.g. for a commanding supervising position within S/C operations away from daily routine (e.g. LEOP activities). The next section describes the implementation of the techniques described above on the example of the Integrated Manual Stack (IMSTK) and its specifications as Master, Echo or Spy.

J. Implementation of GECCOS GUI-Kernel architecture on example of Integrated Manual Stack

The idea of Integrated Manual Stack (IMSTK) arises from an old SCOS NCR and it is supposed to make the manual stack windows easier to locate in a heap of windows on operator screens. The first simple solution was to use a small color icon, which would help to identify one of few manual stacks that the operators have to work with simultaneously during short-time S/C contacts. As GUI parts were re-designed, the new "Integrated Manual Stack" (IMSTK) was built to support multiple manual stack application within a very short session time by integrating a configurable number (for practical reasons 2-6) of Manual Stack GUIs in one window (see Figure 6).

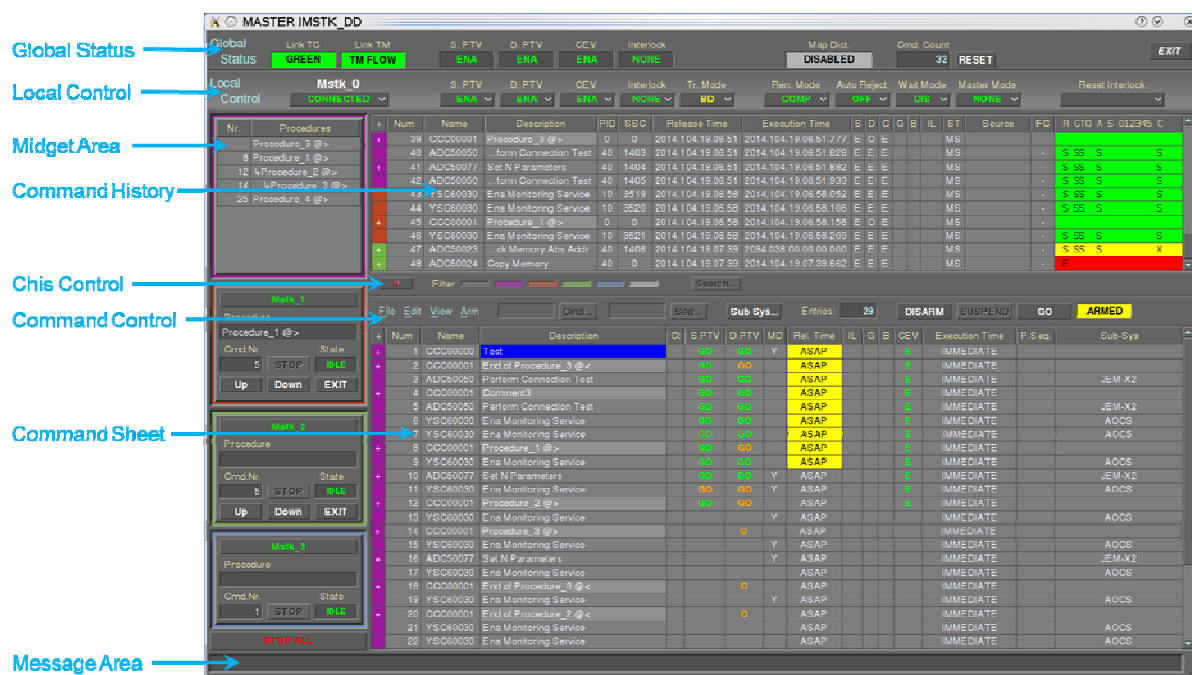


Figure 6 – Integrated Manual Stack. IMSTK integrates four manual stacks and Command History panel on top on basis of well-known ESA SCOS-2000 MSTK paradigm. The stacks are represented by the midgets on the left side.

The IMSTK contains all elements of the "classic" manual stack application as known from SCOS-2000. Additionally, the Command History panel and the miniature panels ("midgets") have been added. The stacks are represented by the midgets on the left side. After start, the first (topmost) Manual Stack is automatically activated. Subsequent stacks can be started on demand by pressing the appropriate midget button; it is also possible to switch between the running stacks, respectively. While only one stack at a time may be active, its content is displayed in the central Command Sheet. The current status of the remaining stacks is displayed within the midgets, as shown in Figure 6. The separate stacks are marked with different colors, and the commands sent by the stacks can be identified by the same color marks within the Command History panel.

In the following we will illustrate on basis of the rather complex Integrated Manual Stack (IMSTK), how two aspects could be minimized: (i) the overall effort and (ii) incorporation of new errors arising from an implementation

from scratch, by utilizing the architectural approach described above (see section H, GECCOS Architecture). Figure 7 B points out how the IMSTK communicates with a number of "BODY" (i.e. kernel sided) parts via D-Bus interface, whereas the functionality of the "BODY" had not been changed, and where new GUI IMSTK is leveled on-top like a shell.

As pointed out above, updates sent by D-Bus can also be registered by "foreign" processes, i.e. foreign IMSTK instances. Following this approach offers the extension of the active Integrated Manual Stack by additional passive "SPY" and "ECHO" roles. The active IMSTK, called "MASTER" sends its requests towards its current BODY's and receives their updates and can in principle be used only by an user with commanding role authorization. Where it is necessary to monitor the state of the "MASTER" commanding stack (e.g. Command Control or -supervisor at Flight Director Position) an IMSTK instance can be started in the passive, so called "SPY" role. In this role it is incidentally supplied with the update signals sent from the appropriate MSTK BODY components (see Figure 7 A for illustration).

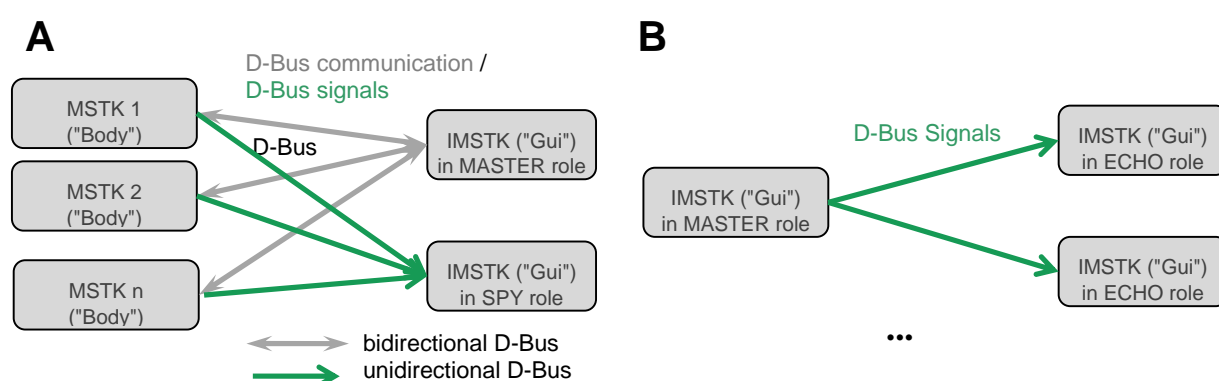


Figure 7 – Applied architecture of GUI-Kernel separation for IMSTK application. Derived runtime modes dependent on D-Bus communication vs. D-Bus signals. A: IMSTK in standard (MASTER) mode and SPY Mode. B: IMSTK in the Master and Echo roles

As a SPY IMSTK instance uses the same update signals that have been sent to the "MASTER", an arbitrary number of IMSTK/SPY can be supported without any additional computational effort on kernel side (i.e. respective BODY parts). Whenever it may be necessary to observe the action of the commanding operator (e.g. big screens in control room during LEOP activities) the signaling capabilities of D-Bus protocol are used as well: One or more instances of IMSTK in "ECHO" role receive the signals emitted by the "Master" (see Figure 7 B). The difference between SPY and ECHO is that ECHO is a pure "echo" of MASTER IMSTK activity, whereas SPY is like a full IMSTK instance without editing and commanding capabilities, running only in a monitoring mode.

The same unidirectional signaling technique as used for IMSTK/ECHO is also used at the integration of the Command History panel within Integrated Manual Stack, now from kernel sided Command Verifier application (i.e. BODY part) to GUI. The server is able to support all running Integrated Manual Stacks with information about the sent commands with minimal computational effort.

K. Automated testing on top (GUI) level

The implementation of modern Qt GUI libraries additionally enabled the introduction of automated tests which are used to conduct comprehensive tests of GECCOS system. For that the GUI test automation tool Squish has been chosen which supports GUI tests for Qt coded applications. Squish allows writing of test scripts with popular scripting languages such as JavaScript, Perl, Python, and Tcl6. Currently implemented GECCOS tests encompass (i) GUI Tests (ii) BODY Tests, (iii) Event Detection Tests (iv) Complex System Tests (v) S/C Data Handling sided Tests.

⁶ The standard Squish functionality has been extended for GECCOS project, by the D-Bus and CORBA Python modules which allow direct communication with selected system components.

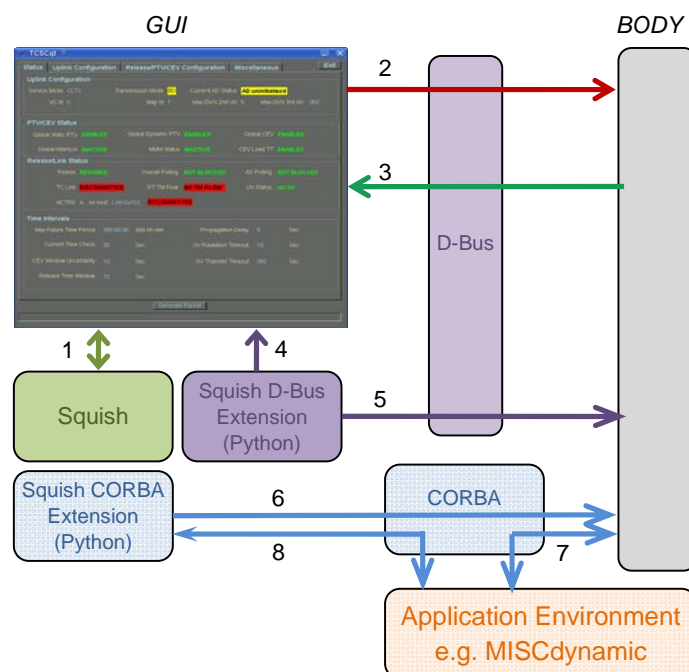


Figure 8 – Application under Test (AUT) for IMSTK: Example of the request – update sequence for various scenarios, as simple GUI update testing (1) or with optionally including D-Bus or CORBA extensions for deep system environment changing tests.

Figure 8 illustrates the possibilities to perform various GECCOS system tests. The simplest scenario is the following: a SQUISH controlled actuator script simulates operator actions and verifies the result on the GUI (Number "1" in figure). This scenario also tests the client-server communication between GUI and BODY (Nr. 2,3). Should the action result in changes of the system environment (7) the effect can be controlled by the CORBA Squish extension module (8). Alternatively the changes can be triggered directly by Squish via D-Bus or CORBA extensions (4,5,6,8). Expected results can be verified either directly (also via extension modules) or on the GUI of the tested application. The possibility to write complex Python test scripts also allows automated mission data system verification, e.g. it is possible to test the whole session MIB (i.e. S/C database), loading and sending predefined saved stack files (SSF) or loading and editing of all MIB commands.

IV. Conclusion

In this work we illustrated how GSOC further developed and improved original ESA SCOS-2000 system into its own system branch, called GECCOS. It is shown how many missions benefit from the concept of a generic Multi-Mission MCS as several fixes and new features can already - or in future - be incorporated to monitoring and control systems (MCS) which are integrated deeply within various mission operation segments (MOS). It profits also from its status as combined MCS and CCS, which is one of the strongest paradigms of the future European generic ground segment data handling kernel, the EGS-CC³. During the modernization phase of GECCOS, several SCOS-2000 originating COTS have been replaced with modern techniques to be prepared for the next decade of mission operations. Especially the new Graphical User Interface (GUI) offers not only a more modern look-and-feel but also automated testing and the use of modern compiler and debugger techniques - in contrast to other SCOS-2000 Systems up to Release 5. Additionally, the GUI has been layered on top of kernel sided processes to separate GUI from data-handling processes. This principle shall additionally allow a more smooth migration in upcoming missions using the new European standard EGS-CC. This will not only avoid new developments and re-trainings, but also the preservation of lots of automated tests which guarantee an optimal validation of new data-handling and processing kernels within integrations of future mission operation segments.

Appendix A Acronym List

AOCS	Attitude and Orbital Control System
COTS	Commercials Of The Shelf
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
ECR	Engineering Change Request
EDRS	European Data Relay Satellite System
EGS-CC	European Ground Systems Common Core
EnMAP	The Environmental Mapping and Analysis Program
FD	Flight Dynamics
G/S	Ground Station
GECCOS	GSOC Enhanced Command and Control System for Operating Spacecrafts
GRACE	Gravity Recovery And Climate Experiment
GSOC	German Space Operations Center
[G]UI	[Graphical] User Interface
H/K	House Keeping [data]
HAG-1	Hispasat Advanced Generation 1
I/F	Interface
IDE	Integrated Development Environment
IMSTK	Integrated Manual Stack
MCS	Monitoring- and Control System (here: used for S/C operations, only)
MIB	Mission Information base (i.e. the S/C database)
MOIS	Manufacturing and Operations Information System (S/W by Rhea)
MOS	Mission Operations Segment
MPS	Mission Planning System
NCR	Non-Conformance Report
PTS	Power Thermal System
PUS	Packet-Utilization Standard (ESA standard)
S/C	Spacecraft
S-MCS	Satellite Monitoring- and Control System (here: used for S/C operations, only)
SLE	Space Link Extension
SSB	SLE Switch Board
S/W	Software
S2K	SCOS-2000
SCOS	Satellite Control and Operation System (SCOS-2000, also SCOS2k)
SmallGEO	Small Geostationary Satellite Platform
TC	Telecommand
TDX	TanDEM-X
TET	Technologie Erprobungs Träger
TM	Telemetry
TSX	TerraSAR-X

Appendix B

Acknowledgements

The implementation work within GECCOS and DLR-SMCS, based on SCOS-2000, was carried out by GSOC's maintenance partner Siemens Convergence Creators, Austria and the company driven by Boguslav Lotko, Austria.

References

- ¹S. Zimmermann, D. Schulze, and C. Stangl: Command Chain Automation. SpaceOps Conference, Pasadena CA, 2014
- ²R. Messaros, C. Stangl, M. Oswald: Evolving Mission Control System Infrastructure for an Altering Fleet of Spacecraft. SpaceOps Conference, Pasadena CA, 2014
- ³A. Walsh, M. Pecchioli, J.M. Carranza, W. Bothmer, P. Schmerber, J. Rueting, P. Parmentier, P. Chirolì, M.C. Charneau, M.P. Geyer et al.: Objectives and Concepts of the European Ground Systems Common Core (EGS-CC). SpaceOps Conference, Stockholm 2012
- ⁴D. Wheeler: More than a Gigabuck: Estimating GNU/Linux's Size. url: <http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>. 2001
- ⁵Consultative Committee for Space Data Systems, CCSDS: CCSDS RECOMMENDATION FOR TELECOMMAND: COMMAND OPERATION PROCEDURES. CCSDS 202.1-B-2, Blue Book. 2001.
- ⁶H. Pennington and D. Wheeler: D-Bus FAQ. URL: <http://dbus.freedesktop.org/doc/dbus-faq.html>, Version 0.3
- ⁷C. Peat and H. Hofmann: SATMON -A Generic User Interface for Satellite Control. DGLR-2004-012